AI Sidecar Command Agent

Experimental MVP – Product Specification & Commercial Prospectus  
Version 1.2 – August 2025

# 1. Executive Summary

This proposal introduces the AI Sidecar Command Agent (AISA), a lightweight local execution framework that extends the utility of AI tools such as ChatGPT. AISA interprets structured command blocks issued by AI and executes them in a secure, sandboxed local environment to manage files, track lineage, and perform Git operations.  
  
While designed as an experimental proof-of-concept MVP, AISA has profound long-term potential for:  
- Enabling AI-assisted version control and publishing  
- Reducing human bottlenecks in coding, writing, and content workflows  
- Introducing traceability and reversibility to AI outputs  
- Forming the basis for compliant, auditable AI-human collaboration systems  
- Seeding future cooperative AI protocols where human intent is structured and validated before execution

# 2. Introduction

Modern AI tools are powerful but stateless—they lack persistent memory, file access, and context awareness across sessions. This tool acts as a local 'sidecar brain' for AI, interpreting structured instructions ([COMMAND] blocks) and handling local file operations, Git interactions, and metadata tracking in a traceable, controllable manner.

# 3. Functional Requirements

|  |  |  |
| --- | --- | --- |
| Requirement | Status | Description |
| Parse [COMMAND] blocks | MVP | Extract structured directives embedded in plain text or .md files. |
| Execute file and Git ops | MVP | Run file operations (create/move/edit/delete) and Git commits. |
| Track lineage/logs | MVP | Maintain changelogs for each operation set. |
| Sandboxed directories | MVP | Ensure all actions remain within a known local structure. |
| Dry-run & rollback | MVP | Simulate commands before execution; enable undo logs. |
| Manual override | MVP | Allow users to approve or modify generated actions. |

# 4. Non-Functional Requirements

- Cross-platform support (Linux, macOS, Windows)  
- Python 3.8+ based; minimal third-party dependencies  
- Configurable safety settings (no external access)  
- Optional telemetry for usage feedback (opt-in)  
- Expandable plugin-based architecture  
- Future GUI/visual interface

# 5. System Architecture

[Diagram Placeholder: Modular architecture diagram showing Command Parser, Shell/Git Executor, Audit Logger, Metadata Tracker (JSON), optional Feedback Agent]

# 6. Build Strategy and Work Estimates

Implementation will be divided into the following phases, using modular testable components:  
- CLI Parser & Sandbox Setup: 6–8 hours  
- Git Command Wrapper (Commit, Pull, Push): 5–7 hours  
- Logging Engine & Dry-run Preview: 5–6 hours  
- JSON Metadata Tracker: 3–4 hours  
- Command Validation & Manual Approval Interface: 4–5 hours  
- Optional GUI (Tkinter or Electron shell): 20–40 hours  
- Feedback Agent or Plugin Protocols: 10–12 hours  
  
Total MVP Timeline: ~25–30 hours (~1–2 developers over a week)

[Remaining sections will be regenerated from prior content and included below.]